Quality of experience driven control of interactive media stream parameters

Bert Vankeirsbilck, Tim Verbelen, Dieter Verslype, Nicolas Staelens, Filip De Turck, Piet Demeester and Bart Dhoedt
Department of Information Technology, Internet Based Communication Networks and Services (IBCN) Ghent University - iMinds
Gaston Crommenlaan 8 Bus 201, B-9050 Gent, Belgium
Email: bert.vankeirsbilck@intec.ugent.be

***Abstract*—In recent years, cloud computing has led to many new kinds of services. One of these popular services is cloud gaming, which provides the entire game experience to the users remotely from a server, but also other applications are provided in a similar manner. In this paper we focus on the option to render the application in the cloud, thereby delivering the graphical output of the application to the user as a video stream. In more general terms, an *interactive media stream* is set up over the network between the user's device and the cloud server. The main issue with this approach is situated at the network, that currently gives little guarantees on the quality of service in terms of parameters such as available bandwidth, latency or packet loss. However, for interactive media stream cases, the user is merely interested in the perceived quality, regardless of the underlaying network situation. In this paper, we present an adaptive control mechanism that optimizes the quality of experience for the use case of a race game, by trading off visual quality against frame rate in function of the available bandwidth. Practical experiments verify that QoE driven adaptation leads to improved user experience compared to systems solely taking network characteristics into account.**

I. Introduction

With the advent of cloud computing, new services such as cloud gaming emerged. In this cloud gaming paradigm, the game is hosted on a server in the cloud. The controls and button presses from the user are transmitted to the server for processing, and the server then sends back the game's response to this user input as a video stream. Examples of popular cloud gaming platforms include onLive and Gaikai. But also other applications than games can be delivered in a similar way. Benefits of this cloud execution approach are mainly related to the fact that the application is not installed on the end user's device, leading to, amongst others, less chance for viruses, low disk space requirements, low-end devices without extreme hardware support needed to support the latest demanding applications and less piracy and illegal copying of applications. A service provider can invest in hardware and optimize the usage by serving multiple users with this hardware to ensure it's return on investment (ROI).

In a simplified model of the cloud computing concept that is valid for this paper, three main parties are involved, namely the user, the network and the cloud service provider. All of these parties have independent goals and requirements. As the service provider typically benefits from serving as many users as possible, a tendency to let the service quality degrade to a sub-excellent level exists. In contrast, the user has less interest in service degradation due to concurrency, and expects qualitative access to the cloud service in proportion to the usage fee paid. The network is unaware of the application level quality, and offers a best effort transport mechanism. It hopes to serve many concurrent connections, and has a limited amount of survival mechanisms such as TCP congestion control at it's disposal. In conclusion, a satisfying level of Quality-of-Experience (QoE) for the user must be reached while operating in a benefit-driven setting for the service provider and using a network that can only offer best-effort Quality-of-Service (QoS).

This paper presents a QoE-driven control system that enables management of cloud application execution quality by optimizing interactive video streaming parameters to user perceived QoE, within the constraints of the underlaying network QoS parameters and the restrictions derived from the service provider's benefit margins. This system bases its decisions on a QoE model of the application being managed, and feedback from monitoring metrics. Section III presents the architecture of the cloud computing system that has been augmented with the proposed control system. As the logic of the controller is centered around models to base its decisions on, Section IV details how the necessary models for the estimation of compression and QoE are acquired. The controller algorithm is presented in Section V. To validate our QoE-aware control mechanism, we applied it to a race game. The results of this experimental validation are described in Section VI. Section VII concludes this paper, including directions for future work.

II. Related Work

In view of the current explosion in cloud-based gaming, a survey of the degradation of the performance due to network latency and packet loss on QoE is presented in [1]. In the field of gaming experience in general, many studies have assessed the effect of network impairments on game experience [2], [3], [4], [5]. In most cases, an indirect application level metric is used to quantify this experience, e.g., the kill/death ratio for a shooter game or lap times in a racing game. This approach requires adaptation to the game (or more generally, the application that is remotely executed), which is not a desired method of operation. Also, such metrics are not generally applicable to other application types, especially when the concept "score" is not well-defined. Alternatives for measuring the interactivity QoE that are not restricted to games consist of logging usage times (departure rate) [6] or mean task execution times [7]. Still, these metrics do not give sufficient information to manage

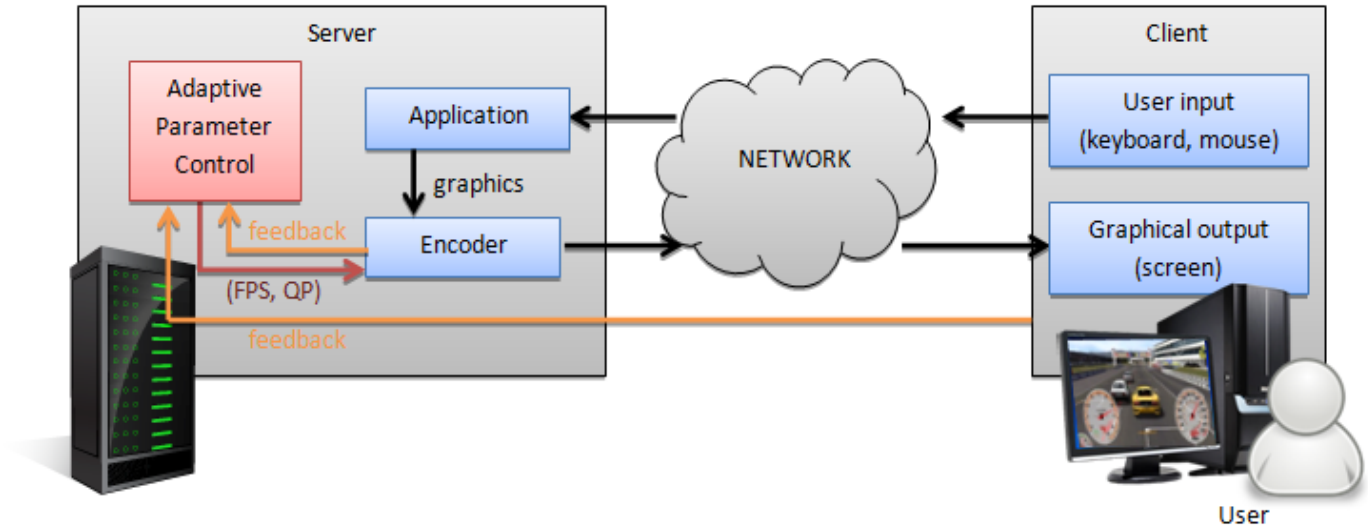


Fig. 1. Architecture of the cloud gaming system augmented with the QoE aware controller. By altering the video stream parameters, the controller optimizes the estimated QoE within the current network circumstances gathered from a monitoring component.

a video stream's parameters in real-time. As a conclusion, we can state that the assessment of interactive media is mostly restricted to games, probably due to the large user base and the demanding nature of these games in comparison to other applications that are either specialized (e.g., Computer Aided Design (CAD) programs) or less demanding concerning resources or graphical behaviour (e.g., office applications).

In [8], motion and scene complexity are measured for various video games within a range of game genres. The main conclusions are that the perspective of a game has an undeniable impact on motion and scene complexity. This discovery shows us that, for our study, it is unlikely to find one model that is suited for all games. Furthermore, the author found that streaming games as a video is only viable for games with very low display resolution and motion. However, we aim to improve the quality of experience by adapting the video stream, to be able to play even more complex games remotely.

Many adaptive controllers for video streams are proposed that are based on the stream-switching technique: the server encodes the video content at different bitrates and it switches from one video version to another based on client feedbacks such as the available bandwidth. For example, in [9], a Quality Adaptation Controller is designed to select the best pre-encoded chunk in a video stream in function of the available bandwidth. Similarly, in [10], a QoE driven selection of such pre-encoded video files is made. Also, in the emerging field of HTTP Adaptive Streaming (HAS), initial work in optimizing QoE using bitrate switching is presented in [11]. In contrast, our case requires real-time encoding of the video content, since the source graphics need to be generated by the application first, making multiple-bitrate encoding unfeasible due to scalability. Hence, on-the-fly adaptation of the parameters of the encoding is demanded to match the network circumstances.

In [12], a QoE driven sender bitrate adaptation scheme at pre-encoding stage is presented that is computes the Peak Signal to Noise Ratio (PSNR) of the delivered media stream. However, in our case, such an approach is practically infeasible, since real-time encoding is required, making pre-encoding impossible. Furthermore, PSNR does not include the QoE-effect of frame rate which, as will be clear further in this paper, is an interesting adaptation parameter for remote application execution. Also, PSNR does not take the difference in user goals into account and hence would not make any difference between the different game genres as was found important in other literature. In [13] a cross-layer adaptation system is made, that controls application bitrate, but also alters the network settings to optimally deliver the video packets. They focus on non-interactive applications such as video playback or file download, as opposed to our case where in essence the interactivity is the main subject of study. The important difference is that for video playback, the quality of the content is the main factor that influences the QoE, while for interactive media the most important factor in the QoE is related to the goal that the user wants to achieve. Depending on this goal, per-frame quality might be secondary to having a crisp response to user actions.

III. Architecture

The architecture of the cloud based gaming system augmented with the QoE aware controller is presented in Fig. 1. At the client side, user input is captures and forwarded to the server for processing. The returned video stream is presented on screen. At server side, the user events are delivered to the game, and the corresponding graphics are captured, and encoded as a video stream. Furthermore, the proposed QoE-aware controller was added, that interfaces with the encoder to alter video streaming parameters in real-time. Examples of such parameters include frame rate (FR) and quantisation parameter (QP). The controller receives feedback about the output bitrate from the encoder. Throughput is measured at the client (e.g., in the form of received bitrate or achieved frame rate) and is communicated to the controller in the server.

Although in Fig. 1, we represent the control cycle of just one connected user, the purpose of cloud computing is actually

serving multiple users from one server. Therefore, in a practical setting, information about the load on the server could also be taken into consideration by the controller, for instance to decide to change the encoder parameters to mitigate server overloads. The architecture presented here is easily extendible to this broader viewpoint. However, for the current paper, we focus on the effect of altering the encoder parameters on the experienced quality by the user, independent of whether the decision to alter the settings is based on network metrics, as is the subject of study in this paper, or on other metrics such as global server load or user subscription type.

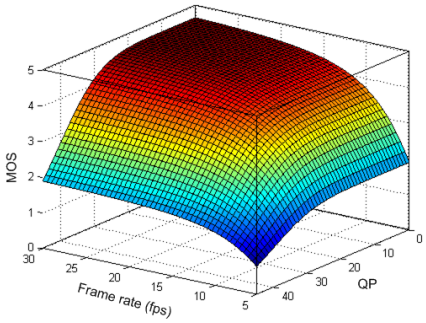
IV. Model Acquisition

QoE is hard to measure at runtime, since this would require input from the user who is supposed to be performing his tasks with the application. Although research exists into the direction of measuring implicit QoE metrics such as facial expressions, heart pulse and blood pressure, we believe the resulting QoE scores are dependent of the total context in which the user is situated, e.g., events in the user's personal life can make facial expression and body parameters fluctuate. Besides, facial recognition is known to be quite computationally intensive and the need to attach and set up additional hardware to control QoE, such as a camera for facial recognition or sensors for heart pulse or blood pressure, is expected to be experienced as too intrusive for the user.

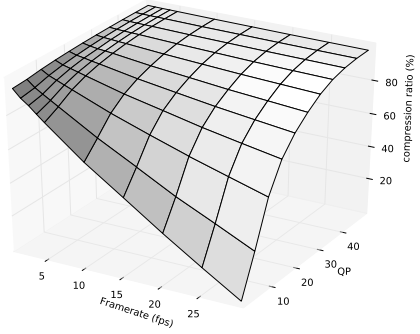
In contrast, we assume a model of the QoE in function of the video stream parameters to be present, specifically for the targeted application. Such a model can be acquired using an evaluation framework developed in previous work [14]. This framework mimics cloud computing on a single machine, and allows to alter network conditions and video streaming parameters in real-time, whilst letting a user interact with a given application, such as the race game *VDrift: Open Source Racing Simulator* [15] in this case. At regular intervals, the user is asked to provide input to a questionnaire about the playability of the game using the current settings, after which new streaming parameters are set for a new evaluation iteration. Applying this method for multiple test users for a given kind of application, a model can be deduced that relates the Mean Opinion Score (MOS) to the relevant streaming parameters. Also, during the tests, the output bitrate of the video streamed application is recorded to create a compression model, that allows to estimate the effect of the various streaming parameters to the output bitrate for the application.

This way, we performed such an assessment with twenty-one test subjects that rated the playability of a race game under varying frame rates and per-frame visual quality settings (controlled via Quantisation Parameter (QP) in the streamer). The models were built for a video stream resolution of 1280×1024. By fitting functions to the acquired data samples, models for QoE and compression were obtained. The QoE model, in its analytical form in (1), was obtained by fitting a four parameter logistic (4PL) function which ensures monotonicity, and relates to the measured data with a coefficient of determination R^2 of 0.95.

$$QoE = \frac{-4.7/(1+(FR/6)^{2.414})+4.7}{1+(QP/42)^6} + 0.04398 \qquad (1)$$



(a) QoE model.



(b) Compression model.

Fig. 2. Models acquired with the subjective test platform, for the VDrift racing game at resolution 1280×1024.

The compression model, with analytic function in (2), is a linear combination of the individual parameter dimensions, yielding a coefficient of determination R^2 of 0.99. It should be noted that the value of QP varies between 0 and 51. The compression is computed relative to the least compressed video format, i.e., a frame rate of *maxFR* (in our tests, the maximum frame rate was defined as 30 fps) and a QP of 0.

$$Compression = 1 - \left[\frac{FR}{maxFR} \times (2^{\frac{-QP}{10}})\right] \qquad (2)$$

Figure 2 presents plots of these functions. Figure 2(a) shows that the QoE model has cut-off thresholds of 15 frames per second (FPS) and a QP of 20 as it comes to playability of the game, meaning that for the race game providing a frame rate above 15 FPS and a QP lower than 20, the user can be expected to be generally satisfied of the quality. If, for some reason, one or both parameters fail to be kept within these boundaries, the user will experience degraded quality, according to the descent presented in the graph. Notice that a MOS of 3 corresponds to *fair quality*, and 2 corresponds to *poor quality* with an *annoying impairment* connotation. Figure 2(b) shows that the compression increases linearly with decreased frame rate, i.e., when one halves the frame rate, the compression (measured over a period of time) logically doubles, as only half of the amount of frames is sent. The QP exhibits an exponential curve.

Combining the two acquired models, as presented in Fig. 3, we conclude that simply choosing settings that result in the most appropriate compression level could lead to suboptimal

QoE. For example, we selected two configurations that demonstrate the difference best. When a frame rate of 6 fps and a QP of 4 is configured, a compression ratio of 84.84% is obtained leading to a MOS of 2.39, denoting a rather *poor* quality would be experienced. However, configuring a frame rate of 18 fps and a QP of 20 yields about the same compression ratio (85%) but results in a MOS of 4.38, which can be interpreted as *good* quality. Namely, for the race game, at these levels of needed compression, it seems that users tend to appreciate a slight drop in both frame rate and per-frame quality better than having frame rate set very low to provide better per-frame quality. Additionally, this figure provides insight into the amount of compression that can be applied without disturbing the user's experience too hard. If therefor a minimum MOS of 3 (*fair* quality) is to be provided, the graph teaches us that configurations exist to guarantee this as long as the needed compression ratio stays below 95.6%. If one targets to provide *good* quality (MOS score higher than 4), the compression ratio is supposed to stay below 91.6%. If the best setting, i.e., QP of 0 and a frame rate of 30 fps, corresponds to a bitrate of 30 Mbps, the respective minimum available bandwidths required are 1.32 Mbps and 2.52 Mbps. However, it must be noted that the user will notice the degraded quality, but the QoE model only describes whether the game is playable or not.

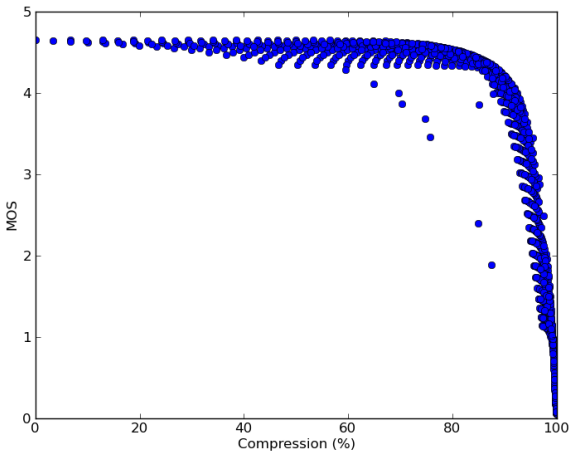


Fig. 3. Depending on the needed compression level, different parameter configurations can be chosen that each lead to a certain MOS. Whenever a choice is to be made amongst different parameter sets, the data represented in this graph can guide the selection of the best tradeoff between QoE and compression.

V. QoE-driven control algorithm

The QoE-driven control algorithm has real-time control over the video stream parameters used in the encoder of the application graphics. As input, it takes the models described higher to estimate the compression and QoE achieved by the various settings it can make, as well as the current output bitrate from the encoder and the feedback of the received bitrate at the client.

The control algorithm, expressed in pseudocode in Algorithm 1 aims to adapt the compression to match the available bandwidth on the network, or a possibly administered bandwidth limit, while keeping user perceived quality as high as possible. First, we propose to load the compression model as a sorted map to be able to easily traverse parameter settings starting from given compression values, as in line 1. It also keeps track of *sampleCounter* that indicates the number of consecutive samples for which the received bitrate is high compared to the encoder output bitrate. To be able to control the speed with which the algorithm responds to better network conditions, the *improveTimer* variable is defined, and is adapted inside the algorithm (lines 2 - 3). Upon receiving a new sample, the algorithm compares the difference between the client received bitrate and the server output bitrate (note that comparing frame rates could make a valid alternative). If the client received bitrate is within a range under the server output bitrate (line 5), the controller interprets this as a network that is able to transfer all of the video stream. In this case, if the *sampleCounter* reached *improveTimer*, the control algorithm tries to alter the parameters to provide higher QoE. First, the QoE improvement mode is activated, by setting the *improveTimer* value to 1, the system can gradually improve QoE at a faster pace (line 7). A step-wise approach is taken, selecting the first lower compression value from the *compressionMap* that gives a better QoE than the current (line 8) and applying it to the encoder (line 9). If the client received bitrate is more than a given threshold under the server output bitrate (line 13), it is assumed that the network can't cope with the video stream in its current quality. The *sampleCounter* and *improveTimer* are reset to back off improving the QoE (lines 14 and 15). Higher compression is to be applied to stay under the client received bitrate, as the latter is considered an indication of what the network can cope at that moment. The amount of compression needed is computed in line 18. Then, the algorithm searches for the pair of frame rate and QP that gives the highest QoE with a compression value higher than the needed target compression (lines 19 - 24), and applies it to the encoder (line 25).

This algorithm responds fast to degraded network conditions, and waits for the network to stabilize before trying to improve the QoE again. This approach is necessary since the decisions made upon the measurements of the network conditions actually come down to deciding to act at a given time to a network situation in the recent past. Hence, network degradations should be acted upon immediately to accelerate its recovery, while to improve the quality of the service, careful, stepwise initiatives should be taken.

VI. Experimental validation

With the framework used for the model acquisition, described in Section IV, we executed the VDrift game [15] for which the models were derived earlier. This framework is installed on a server with an Intel®Core™i7 CPU 920 @ 2.67 GHz, 6 GB RAM Graphics Card NVidia GeForce GTS 250, 128 CUDA cores with Ubuntu Linux operating system. The game server software, with the game execution environment, the video encoder and the QoE-driven controller are co-located with the viewer software that decodes the video stream and captures the user input, using the localhost network to transmit the user events and the video stream. This approach implies having more stable control over the network parameters compared to executing viewer and server on separate machines. The Linux traffic control package (*tc* and *tcng*) can be used to introduce e.g., packet loss, bandwidth drops or latency. For

Algorithm 1 QoE-driven control algorithm

$compressionMap = loadCompressionModel()$
$sampleCounter = 0$
$improveTimer = 1$
while $running$ **do**
  if $Bitrate_{received} \geq Bitrate_{output} - threshold$ **then**
    if $sampleCounter \geq improveTimer$ **then**
      $improveTimer = 2$
      $(FR, QP)_{improved}$ (first lower compression with higher QoE)
      $encoder.apply(FR_{improved}, QP_{improved})$
    else
      $sampleCounter++$
    end if
  else
    $sampleCounter = 0$
    $improveTimer = 5$
    $FR_{adapted} = FR_{current}$
    $QP_{adapted} = QP_{current}$
    $compression_{target} = \frac{Bitrate_{received}}{[\frac{Bitrate_{output}}{(1-compression_{current})}]}$
    for all $(FR, QP) \in \{(FR, QP)_{>compression_{target}}\}$ **do**
      if $QoE(FR, QP) > QoE(FR_{adapted}, QP_{adapted})$ **then**
        $FR_{adapted} = FR$
        $QP_{adapted} = QP$
      end if
    end for
    $encoder.apply(FR_{adapted}, QP_{adapted})$
  end if
end while

these experiments, the network degradations were constrained to bandwidth variations. A video resolution of 1280×1024 was configured, matching the created models.

Figure 4 shows how the QoE-aware controller manages to provide higher QoE over other approaches that keep either frame rate or QP factor fixed while varying the other. The chosen fixed frame rates were 30 fps and 25 fps, the fixed QP values were chosen at 0 and 5. These values represent the top two least degraded values of frame rate and visual quality that the QoE-aware controller takes into account. It can also be seen that for the race game targeted in this use case, keeping the per-frame quality high (setting a low QP value) while degrading the frame rate to achieve compression performs worse than focusing on maintaining a high frame rate. Our algorithm always aims to find a trade-off between the two parameters, hence provides the optimal quality for the current network bandwidth.

Figure 5(a) reports how the algorithm reacts to bandwidth variations that were deployed using a shaper. The blue line indicates the available bandwidth in the network configured with this shaper, the red line indicates the output bitrate of the encoder that is achieved with the QoE aware controller. Although large fluctuations in the configured bandwidth exist, the algorithm succeeds in gracefully adapting to the available bandwidth. In the time intervals where large bandwidth increases exist (55 to 60 seconds and 80 to 120 seconds), it is visible that a conservative, stepwise approach to increasing

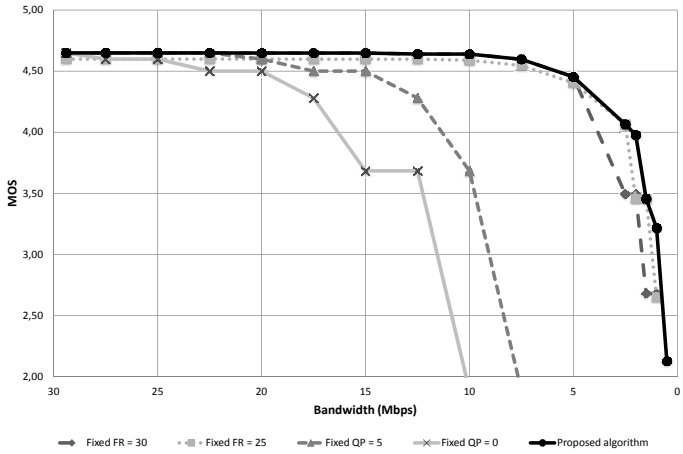


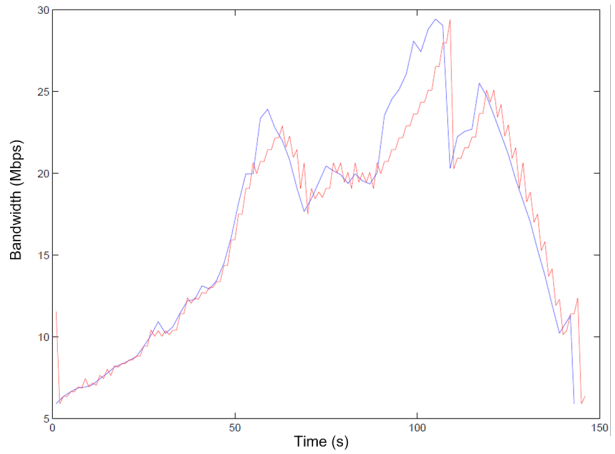
Fig. 4. Comparison of the QoE obtained with the QoE-aware control algorithm, versus simpler approaches that keep either frame rate or QP fixed and varying the other parameter to get the necessary compression.

the QoE is taken. On the other hand, bandwidth drops are handled more aggressively, as visible in the graph at times about 120 seconds and the steep descending behaviour of the bandwidth at the end of the test. As the monitoring values are only communicated to the controller *after* bandwidth changes have occured in the network, we see that for bandwidth drops the corrective decision always falls one sample behind at least. However, since the available bandwidth cannot be easily measured without overloading the network, one is restricted to a reflective approach as taken in the proposed algorithm.

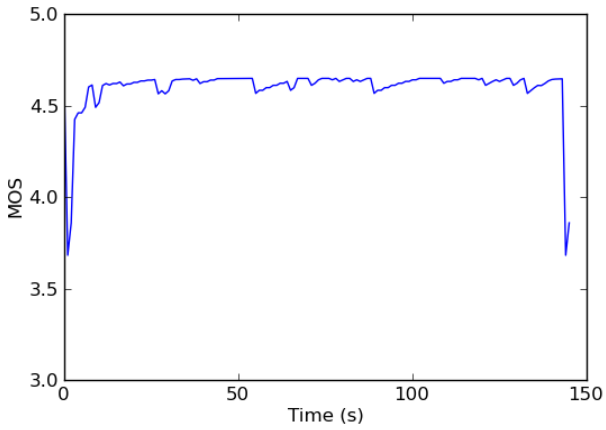
Figure 5(b) shows the QoE achieved during the test. It shows that, despite the fluctuations in the bandwidth, the QoE is high, and fairly constant. However, it must be noted that, although real-time updating of the encoder parameters is guaranteed, a stepwise changing of the encoder parameters will be visible to the user. This graph does not provide information on how this frequent quality changing is experienced by the user, and it is possible that the user would prefer a more constant, beit a possibly lower quality, over highly frequent updates of the encoder settings.

VII. Conclusion

In this paper a QoE-driven control system is proposed that enables management of cloud gaming quality by, in real-time, optimizing interactive video streaming parameters to an estimation of the user perceived QoE, within the constraints of the underlaying network QoS parameters and possibly the restrictions expressed from the service provider to allow scaling the service for a large user base. This system bases its decisions on different QoE models for the various game genres offered in the service, and feedback from monitoring metrics. The acquisition of the QoE models is performed offline with a selected test public, using a dedicated subjective evaluation framework. Although our study focused on one game type, literature indicates that different QoE models will be found depending on the game genre, e.g., for shooter games it is expected that frame rate will be found more important for the QoE than per-frame visual quality, while for a rather slow-paced text based game the readability of the graphics demand priority over frame rate. Upon receiving samples of



(a) Reaction of the controller to bandwidth variations deployed with a shaper. The blue line represents the available bandwidth (i.e., the target of the shaper), the red line indicates the encoder output bitrate.



(b) The QoE achieved in the course of the bandwidth variation.

Fig. 5. Validation experiment results using bandwidth shaper

the received bitrate or frame rate from the viewer, the control algorithm decides whether to downgrade the video streaming quality to mitigate network degradations or to try to upgrade the quality to provide better experience to the user. This control algorithm responds fast to degraded network conditions, and waits for the network to stabilize before trying to improve the QoE again. Experimental validation results obtained with an implementation of the controller while managing the playability of a race game are presented. The results show that the controller optimizes the QoE to the network circumstances better than naive approaches to mitigate network degradations. However, although theoretically the optimal QoE is aimed for at any moment, it should be investigated whether it is better to stay with suboptimal settings for a while over switching qualities frequently. We expect that the proposed algorithm might need an additional QoE parameter in the models, i.e., to avoid overly frequent stream quality switching, which is a possibility to study in more detail.

Future work also includes acquiring and comparing additional models for different game genres and even non-game applications, to verify the ability of the proposed controller to manage the QoE of arbitrary applications in the cloud. It would also be interesting to thouroughly evaluate the fairness while executing multiple controlled applications, and the QoE achieved when the system contends with an unmanaged application. Finally, subjective assessment with test persons playing the game managed by the QoE-driven controller over a longer period of time would be an important improvement to evaluate and demonstrate the potential of the work presented in this paper.

REFERENCES

[1] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hossfeld, "Gaming in the clouds: QoE and the users' perspective," *Mathematical and Computer Modelling*, 2012.

[2] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of factors affecting players' performance and perception in multiplayer games," in *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, ser. NetGames '05. New York, NY, USA: ACM, 2005, pp. 1–7.

[3] P. Quax, P. Monsieurs, W. Lamotte, D. De Vleeschauwer, and N. Degrande, "Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game," in *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, ser. NetGames '04. New York, NY, USA: ACM, 2004, pp. 152–156.

[4] M. Ries, P. Svoboda, and M. Rupp, "Empirical study of subjective quality for massive multiplayer games," in *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*, june 2008, pp. 181–184.

[5] A. F. Wattimena, R. E. Kooij, J. M. van Vugt, and O. K. Ahmed, "Predicting the perceived quality of a first person shooter: the Quake IV G-model," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, ser. NetGames '06. New York, NY, USA: ACM, 2006.

[6] K.-T. Chen, P. Huang, and C.-L. Lei, "How sensitive are online gamers to network quality?" *Commun. ACM*, vol. 49, no. 11, pp. 34–38, Nov. 2006.

[7] S. H. Yoo and W. C. Yoon, "Modeling users' task performance on the mobile device: PC convergence system," *INTERACTING WITH COMPUTERS*, vol. 18, no. 5, pp. 1084–1100, sept 2006.

[8] M. Claypool, "Motion and scene complexity for streaming video games," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, ser. FDG '09. New York, NY, USA: ACM, 2009, pp. 34–41.

[9] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proceedings of the second annual ACM conference on Multimedia systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 145–156. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943573

[10] A. Khan, L. Sun, E. Jammeh, and E. Ifeachor, "Quality of experience-driven adaptation scheme for video applications over wireless networks," *Communications, IET*, vol. 4, no. 11, pp. 1337 –1347, June 2010.

[11] V. Menkovski and A. Liotta, "Intelligent control for adaptive video streaming," in *In Proceedings of the International Conference on Consumer Electronics*, Las Vegas, US, January 2013.

[12] A. Khan, L. Sun, E. Ifeachor, J. Fajardo, and F. Liberal, "Video quality prediction model for h.264 video over umts networks and their application in mobile video streaming," in *Communications (ICC), 2010 IEEE International Conference on*, may 2010, pp. 1 –5.

[13] S. Thakolsri, S. Khan, E. Steinbach, and W. Kellerer, "Qoe-driven cross-layer optimization for high speed downlink packet access," *Journal of Communications*, vol. 4, no. 9, 2009.

[14] B. Vankeirsbilck, D. Verslype, N. Staelens, P. Simoens, C. Develder, P. Demeester, F. D. Turck, and B. Dhoedt, "Platform for real-time subjective assessment of interactive multimedia applications," *Multimedia Tools and Applications*, 2013, 10.1007/s11042-013-1395-y.

[15] VDrift, "VDrift Open Source Racing Simulator, version 2011-10-22," http://www.vdrift.net.